OpenCUDA+MPI

A Framework for Heterogeneous GP-GPU Cluster Computing

Kenny Ballou

March 20, 2013

1 GP-GPU Computing Introduction

2 Problems and Solutions

3 A Little About Methodology

4 What's Been Done

1 GP-GPU Computing Introduction
- Parallel versus Distributed Computing
- Applications of Supercomputing
- Who Uses Supercomputing?

2 Problems and Solutions

3 A Little About Methodology

4 What's Been Done

Introduction

Parallel and Distributed Computing

What is GP-GPU Distributed Computing?

- Parallel:
  - Processing concurrently
- Distributed:
  - Processing over many computers, typically in parallel, but not always
  - Local
  - Grid Computing

Applications of Supercomputing

What can we do with Parallel and Distributed Computing?

- Solving (Large) Linear Systems
  - LINPACK Benchmarks
- Fluid Dynamic Simulations
- N-Body Simulations
- Brute-Force Password/Hash Cracking
- Prime Number Searching
- Protein Folding
- Image Analysis / Manipulation
- . . .

Who Uses Supercomputing?

ALL The People

Who Uses Supercomputing?

No Really...

- Google – Page Indexing
  - Created Map-Reduce
- Facebook – Data Mining
- Universities
- Many Others

1 GP-GPU Computing Introduction

2 Problems and Solutions
- Problems with Current Solutions
- Solutions
- Plans and Goals

3 A Little About Methodology

4 What's Been Done

Problems

- "Distributed Programming" is expensive
- Specificity of Hardware
- Data
  - Distribution
  - Volume
  - NFS
- Fault Tolerance
- Optimizing Resources and Utilization

A Framework

Solutions

- Ease Programming Interface for Highly Parallel Distributed Computing
- Allow for Diversity in Computing Environment
  - Bring together ideas from both types of distributed computing
  - "Jungle Computing"

Plan and Goals

- Develop a framework for distributed computing over a heterogeneous cluster
- Develop several different solutions for vascular extraction from CT angiography scans
- Profile the different solutions
- Add Cluster/ Node Configuration and Scheduling Options
- Release as FOSS to the world

1 GP-GPU Computing Introduction

2 Problems and Solutions

3 A Little About Methodology
- Implementation Details

4 What's Been Done

Implementation Details

- Arch Linux
- Salt
- Python
- CUDA
- (Open)MPI

Arch Linux

- Core Tenet: Minimalism
- Small
- Lightweight
- Familiarity

Salt

More than just NaCl

Provisioning tool for managing infrastructure

- Allows for "Push" based state changes
- Remote Execution
- Simplicity
- Fast

Python

- Development Speed: Expressive and Readable
- Fast Enough
  - Written in `C/C++`
- Great and Many Profiling Tools
  - `time.time()`
  - `timeit`
  - `cProfile`
  - `...`
- Where slow, allows use of `C/C++` code

CUDA

Compute Unified Device Architecture

- Established interface with (nVidia) GPU's
- `pyCUDA`
  - Deferred CUDA kernel compilation
- Familiarity

(Open)MPI

- Established interface for inter-process communication
- `mpi4py`
  - One of the most complete MPI implementations

1 GP-GPU Computing Introduction

2 Problems and Solutions

3 A Little About Methodology

4 What's Been Done
- What I have been doing
- Moving Forward
- Problems Encountered
- Potential Solutions

Tasks

- Learning MPI, `mpi4py`, and `pyCUDA`
- Node/ Cluster Administration
  - Node Build Scripts
  - Salt Configuration
    - Special Thanks to Danny
- *Lots* of thinking

Moving Forward

- Finish Creating Salt States and Configuration
- Continue Learning `MPI` and `pyCUDA`
- Develop Framework

Problems / Roadblocks

- Power Requirements
- NFS Share `/home` performance
- Time?

Potential Solutions

- Request(ing) more suitable and stable power
- Researching Distributed Filesystems / File storage

OpenCUDA+MPI

A Framework for Heterogeneous GP-GPU Cluster Computing

Kenny Ballou

March 20, 2013